Previously

• Resources aren’t free
• Share to reduce costs
• Schedule operations on resources
  – Fixed resources
• Greedy approximation algorithm
• List Scheduling for resource-constrained scheduling

Today

• Tighter bounds on List Scheduling
• Time-Constrained Scheduling
  – Force Directed
• Few words on project architecture
• Resource-Constrained
  – Branch-and-Bound

Precedence Constrained

• Optimal Length > All busy times
  – Optimal Length ≥ Resource Bound
  – Resource Bound ≥ All busy
• Optimal Length > This Path
  – Optimal Length ≥ Critical Path
  – Critical Path ≥ This Path
• List Schedule = This path + All busy times
• List Schedule ≤ 2 *(Optimal Length)

Conclude

• Scheduling of identical parallel machines with precedence constraints has a 2-approximation.
Tightening

• How could we do better?

• What is particularly pessimistic about the previous cases?
  – List Schedule = This path + All busy times
  – List Schedule ≤ 2 *(Optimal Length)

Tighten

• LS schedule ≤ Critical Path+Resource Bound
• LS schedule ≤ Min(CP,RB)+Max(CP,RB)
• Optimal schedule ≥ Max(CP,RB)
• LS/Opt ≤ 1+Min(CP,RB)/Max(CP,RB)

• The more one constraint dominates ➔ the closer the approximate solution to optimal
  % (EEs think about 3dB point in frequency response)

Tightening

• Example of
  – More information about problem
  – More internal variables
  – …allow us to state a tighter result
• 2-approx for any graph
  – Since CP may = RB
• Tighter approx as CP and RB diverge

Multiple Resource

• Previous result for homogeneous functional units
• For heterogeneous resources:
  – also a 2-approximation
    • Lenstra+Shmoys+Tardos, Math. Programming v46p259
    • (not online, no precedence constraints)

Bounds

• Precedence case, Identical machines
  – no polynomial approximation algorithm can achieve better than 4/3 bound
    • (unless P=NP)
• Heterogeneous machines (no precedence)
  – no polynomial approximation algorithm can achieve better than 3/2 bound

Preclass
Preclass

- Critical Path LB?
- Resources to keep RB < CP?
- Resources to achieve CP?
  - Take poll: 4, 3, 2, 1
- What was trick to achieving?
- Why might List Schedule have a problem with this?

Force Directed

**Problem:** how exploit schedule freedom (slack) to minimize instantaneous resources

- Directly solve time-constrained scheduling
  - (previously only solved indirectly)
  - Minimize resources with timing target

If everything were scheduled, except for the target node, *what would we do?:*

- examine resource usage in all timeslots allowed by precedence
- place in timeslot that has least increase maximum resources
  - Least energy
  - Where the forces are pulling it
**Force-Directed**

- **Problem:** don't know resource utilization during scheduling
- **Strategy:** estimate resource utilization

**Force-Directed Estimate**

- Assume a node is uniformly distributed within slack region
  - between earliest and latest possible schedule time
- Use this estimate to identify most used timeslots

**Single Resource Challenge**

Schedule into 12 cycles

**Slacks on all nodes**

In order to estimate, will need to break each task into fractions. With slack 8, can go in any of 9 slots. With slack 2, can go into any of 3 slots.
Force-Directed

- Scheduling a node will shift distribution
  - all of scheduled node’s cost goes into one timeslot
  - predecessor/successors may have freedom limited so shift their contributions
- **Goal**: shift distribution to minimize maximum resource utilization (estimate)
Many steps…
Force-Directed Algorithm

1. ASAP/ALAP schedule to determine range of times for each node
2. Compute estimated resource usage
3. Pick most constrained node (in largest time slot…)
   - Evaluate effects of placing in feasible time slots (compute forces)
   - Place in minimum cost slot and update estimates
   - Repeat until done

Force-Directed Runtime

- Evaluate force of putting in timeslot $O(N)$
  - Potentially perturbing slack on net prefix/postfix for this node $\rightarrow N$
- Each node potentially in $T$ slots: $\times T$
  - $T =$ schedule target
- $N$ nodes to place: $\times N$
- $O(N^2T)$
  - Loose bound--don’t get both $T$ slots and $N$ perturbations

Branch-and-Bound

(for resource-constrained scheduling)
Brute-Force Scheduling (Exhaustive Search)

- Try all schedules
- Branching/Backtracking Search
- Start w/ nothing scheduled (ready queue)
- At each move (branch) pick:
  - available resource time slot
  - ready task (predecessors completed)
  - schedule task on resource
  - Update ready queue

Example

![Example Diagram]

Branching Search

- Explores entire state space
  - finds optimum schedule
- Exponential work
  - \( O(N^{\text{resources} \times \text{time-slots}}) \)
- Many schedules completely uninteresting

Reducing Work

1. Canonicalize "equivalent" schedule configurations
2. Identify "dominating" schedule configurations
3. Prune partial configurations which will lead to worse (or unacceptable results)

"Equivalent" Schedules

- If multiple resources of same type
  - assignment of task to particular resource at a particular timeslot is not distinguishing

...
“Non-Equivalent” Schedule Prefixes

Prefixes

Pruning Prefixes

• Keep track of scheduled set
• Recognize when solving same sub-problem
  – Like dynamic programming finding same sub-problems
  – But no guarantee of small number of subproblems
    • set is power-set so $2^N$
    • …but not all feasible,
    – so shape of graph may simplify

Pruning

• If can establish a particular schedule path will be worse than one we’ve already seen
  – we can discard it w/out further exploration
• In particular:
  – LB=current schedule time + lower_bound_estimate
  – if LB greater than known solution, prune

Pruning Techniques

Establish Lower Bound on schedule time
• Critical Path (ASAP schedule)
• Resource Bound

Dominant Schedules

• A strictly shorter schedule
  – scheduling the same or more tasks
  – will always be superior to the longer schedule

Alpha-Beta Search

• Generalization
  – keep both upper and lower bound estimates on partial schedule
    • Lower bounds from CP, RB
    • Upper bounds with List Scheduling
  – expand most promising paths
    • (least upper bound, least lower bound)
    • prune based on lower bounds exceeding known upper bound
    – (technique typically used in games/Chess)
Alpha-Beta

- Each scheduling decision will tighten
  - lower/upper bound estimates
- Can choose to expand
  - least current time (breadth first)
  - least lower bound remaining (depth first)
  - least lower bound estimate
  - least upper bound estimate
- Can control greediness
  - weighting lower/upper bound
  - selecting “most promising”

Note

- Aggressive pruning and ordering
  - can sometimes make polynomial time in practice
  - often cannot prove will be polynomial time
  - usually represents problem structure we still need to understand
- Coudert shows scheduling
  - Exact Coloring of Real-Life Graphs is Easy”, in Proc. of 34th DAC, Anaheim, CA, June 1997.

Multiple Resources

- Works for multiple resource case
- Computing lower-bounds per resource
  - resource constrained
- Sometimes deal with resource coupling
  - e.g. must have 1 A and 1 B simultaneously or in fixed time slot relation
    - e.g. bus and memory port

Summary

- Resource estimates and refinement
- Branch-and-bound search
  - “equivalent” states
  - dominators
  - estimates/pruning

Project Architecture

FPGA Architecture
FPGA Lookup Tables (LUTs)

- Common mux failure: cannot switch 0, 1
  - Can hold a constant 0 or 1 output

Fully Functional LUT

- Compute $A \cdot B \cdot C \cdot D$ with:

Using Partially Defective LUT

- Compute $A \cdot B \cdot C \cdot D$ with:

Big Ideas:

- Estimate Resource Usage
- Use dominators to reduce work
- Techniques:
  - Force-Directed
  - Search
    - Branch-and-Bound
    - Alpha-Beta
Admin

• Assignment 1 was due at class start
• Reading
  – Wednesday online
• Assignment 2 out
  – Part A due next Monday
  – Part B following